네트워크기반 병렬 유전자 알고리즘을 이용한 중앙집중형 동적부하균등기법의 성능향상

송봉기*, 성길영**, 우종호***

Performance Improvement of Centralized Dynamic Load-Balancing Method by Using Network Based Parallel Genetic Algorithm

Bong Gi Song* , Kil Young Sung**, Chong Ho Woo***

요 약

본 논문에서는 중앙집중형 동적부하균등을 효율적으로 처리하기 위하여 네트워크기반 병렬 유전자 알고리즘을 이용하였다. 기존의 유전자 알고리즘을 적용한 경우와는 달리 클라이언트들에서 최적작업 할당의 탐색을 분산처리하여 중앙 스케줄러의 성능을 향상시킬 수 있었다. 최적해의 수렴속도를 향상시키기 위해 선택연산은 룰렛휠 선택과 엘리트 보존전략을 함께 사용하였고, 염색체 인코딩은 슬라이딩윈도우기법을 이용하였으며 교차연산은 주기교차방법을 이용하였다. 부하균등기법의 유연성 변화에 따른 중앙 스케줄러의 성능을 모의실험한 결과 기존의 방법보다 성능이 향상됨을 확인하였다.

Abstract

In this paper, the centralized dynamic load-balancing was processed effectively by using the network based parallel genetic algorithm. Unlike the existing method using genetic algorithm, the performance of central scheduler was improved by distributing the process for the searching of the optimal task assignment to clients. A roulette wheel selection and an elite preservation strategy were used as selection operation to improve the convergence speed of optimal solution. A chromosome was encoded by using sliding window method. And a cyclic crossover was used as crossover operation. By the result of simulation for the performance estimation of central scheduler according to the change of flexibility of load-balancing method, it was verified that the performance is improved in the proposed method.

키워드

병렬 유전자 알고리즘, 동적부하균등기법, 작업할당, 중앙 스케줄러

Ⅰ. 서 론

동적부하균등기법은 새로운 태스크가 시스템에 도착할 때 작업할당계획을 수립하고 그 계획에 따라 작업을 할당함으로써 전체 태스크 수행시간을 최소화하고 프로세서의 활용도를 최대화하는 기법이다.[1-3] 태스크가 수시로 발생하고 그 수행시간을 예측하기 어려운 경우, 빠르고 효과적인 작업할당이 요구된다. 이를 위해 동적부하균등기법의 작업할당계획을 수립시 유전자 알고리즘(genetic algorithm)을 적용하여 처리시간을 단축시키고 최적의 작업할당계획을 얻고자 하는 연구가 있어왔다.[4-7] M. Munetomo[6] 등은 병렬 분산 시스템 환경에서 유전자 알고리즘을 이용한 부하균등기법을 제안하였다. 그러나, 분산형 동적부하균등기법을 이용하여 프로세서간 통신오버헤드가 크다. A.

* 부경대학교 정보시스템
** 경상대학교 정보통신공학과
*** 부경대학교 전자컴퓨터정보통신공학부
접수일자 : 2004. 10. 20

Y. Zomaya[4] 등은 중앙집중형 동적부하균등기법에 유전자 알고리즘을 적용하는 방법을 제안하였다. W. A. Greene[7]은 다양한 확률분포로 작업이 도착하는 동적부하균등에 유전자 알고리즘을 이용하는 방법을 제안하고 각 확률분포에 따른 성능을 평가하였다.

본 논문에서는 중앙 스케줄러(central scheduler)에서 네트워크기반 병렬 유전자 알고리즘(NBPGA: Network Based Parallel Genetic Algorithm)을 이용하여 최적작업할당의 탐색을 분산처리하여 중앙 스케줄러의 성능을 향상시켰다. 중앙 스케줄러에서 생성된 염색체를 클라이언트들에 분할하여 처리하고, 클라이언트에서 구해진 지역 최적해들을 서버가 유휴시간에 개선함으로써 효율적인 작업할당계획을 수립하였다. 이 때 염색체 인코딩은 슬라이딩윈도우기법(sliding window method)을 이용하고, 선택연산은 룰렛휠 선택연산과 엘리트연산을 함께 이용하며 교차연산과 돌연변이연산은 각각 주기교차방법과 임의의 두 점을 택해 값을 교환하는 돌연변이방법을 이용한다. 이렇게함으로써 기존의 방법보다 고속처리가 가능한 작업계획을 구할 수 있다.

본 논문의 구성은 2장에서 중앙집중형 동적부하균등기법에 대해 살펴보고 3장에서 네트워크기반 병렬 유전자 알고리즘 관련 연구들을 살펴본다. 4장에서 중앙집중형 동적부하균등기법에 네트워크기반 병렬 유전자 알고리즘을 이용하는 방법을 설명한다. 5장에서는 실험을 통해 성능을 평가 및 고찰하며 6장에서 결론을 맺는다.

II. 중앙집중형 동적부하균등기법

부하균등기법은 프로세서의 활용도를 최대화하기 위해 프로세서들에 부하를 균등하게 할당하는 기법이다. 이를 위해 각 프로세서의 부하정보를 지속적으로 갱신하고 작업할당을 신속하게 결정해야 한다. 부하균등기법은 그 특성에 따라 중앙집중형과 분산형, 동적과 정적 등으로 분류할 수 있다.[4,8-10]

중앙집중형 부하균등기법과 분산형 부하균등기법의 특징은 다음 표 1과 같다. 중앙집중형 부하분산기법은 중앙 스케줄러라는 단일프로세서를 이용하여 전역부하정보(global load information)를 수집하고 부하할당계획을 수립한다. 분산형 부하균등기법은 시스템 내의 각 프로세서가 자신의 부하정보를 다른 프로세서들에 방송하고 지역적으로 수정된 부하정보를 갱신한다. 시스템 내의 모든 프로세서들이 전역부하정보를 유지하므로 각 프로세서들이 부하할당을 결정할 수 있다. 중앙집중형 부하균등기법은 분산형 부하균등기법보다 프로세서간 통신 오버헤드가 작기 때문에 대형시스템에 적합하다. 그러나 중앙 스케줄러의 실패가 전체 부하균등정책의 기능장애를 초래하므로 신뢰성이 떨어진다.[9]

표 1. 중앙집중형 부하분산기법과 분산형 부하분산기법의 비교
Table 1. The comparison of centralized load-balancing method and decentralized load-balancing method

	중앙집중형 부하분산기법	분산형 부하분산기법
부하정보 관리자	중앙 스케줄러	각 프로세서
장점	프로세서간 통신 오버헤드가 작다.	신뢰성이 높다.
단점	신뢰성이 낮다.	프로세서간 통신 오버헤드가 크다.

정적부하균등기법에서 작업할당은 컴파일 시간에 결정된다. 반면 동적부하균등기법에서는 새로운 태스크가 시스템에 도달할 때 결정된다. 동적부하균등기법은 일반적으로 정적 부하균등기법보다 구현이 어렵지만 더 효과적인 부하균등이 가능한 특징을 가진다. 정적 부하균등기법에서는 특정 프로세서에 태스크가 편중되어 할당될 수도 있으며 이로 인하여 전체 시스템의 성능이 저하된다.

중앙집중형 동적부하균등기법에서 전체 시스템의 성능은 중앙 스케줄러의 성능에 의존하므로 중앙 스케줄러는 효율적인 작업할당을 신속하게 결정해야 한다. 이를 위해 A. Y. Zomaya[4] 등은 유전자 알고리즘을 이용하는 방법들을 연구하였다. 유전자 알고리즘은 생물의 유전시스템을 모방하여 개체군에 선택, 교차, 돌연변이 연산들을 반복적으로 적용하여 근사 최적해를 구하는 다점검색기법이다.[11] Zomaya 등은 슬라이딩윈도우기법을 이용하여 염색체 인코딩하고 주기교차연산 및 전형적인 돌연변이연산을 이용하여 작업할당계획을 수립하는 모델을 제안하였다.

III. 네트워크기반 병렬유전자 알고리즘

유전자 알고리즘에서 계산속도의 향상을 위해

유전자 알고리즘의 병렬화에 대한 연구가 이루어졌다. 그러나 대부분의 병렬 유전자 알고리즘들은 특수한 병렬컴퓨팅 환경을 요구하므로 비용이 증가하는 단점이 있다. 이런 단점을 개선하여 기존의 네트워크 환경을 이용하여 유전자 알고리즘의 병렬화를 가능하도록 하는 것이 네트워크기반 병렬 유전자 알고리즘이다.[12-14]

네트워크기반 병렬 유전자 알고리즘은 그림 1과 같이 서버/클라이언트모델을 기반으로 구성된다. 서버는 초기화 프로세스(I), 버퍼(B), 유전자 알고리즘(GA_S), 엘리트 풀(P)로 구성되고 클라이언트는 이주관리자(MM)와 유전자 알고리즘(GA_C)로 구성된다. 이 때 서버의 유전자 알고리즘과 클라이언트의 유전자 알고리즘은 동일한 평가함수와 유전연산자를 이용한다. 그러나, 개체군의 크기는 서버의 유전자 알고리즘과 클라이언트의 유전자 알고리즘에서 서로 상이하다. 서버의 유전자 알고리즘에서 개체군의 크기는 클라이언트의 수 + 1개로 제한되는 반면 클라이언트의 유전자 알고리즘은 개체군의 크기가 제한되지 않는다.

초기화 프로세스는 클라이언트에 염색체를 전달하고 서버 및 클라이언트의 유전자 알고리즘을 초기화한다. 버퍼는 클라이언트로부터 전달된 지역 최적해를 저장하고 엘리트 풀은 서버에서 생성된 전역 최적해를 저장한다. 서버의 유전자 알고리즘은 클라이언트에서 전달된 지역 최적해들로 개체군을 형성하여 전역 최적해를 구한다.

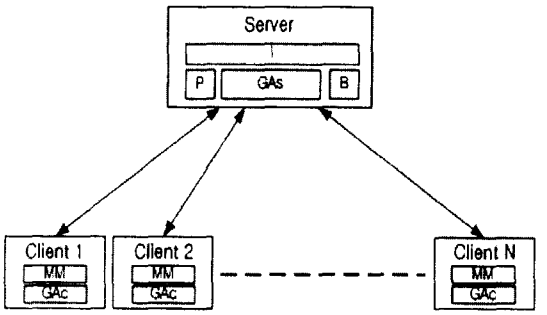


그림 4. 서버의 계산능력을 활용한 네트워크기반 병렬 유전자 알고리즘
Fig 1. The network based parallel genetic algorithm by exploiting server's computing power

이주관리자는 클라이언트에서 서버로의 지역 최적해 이주와 서버에서 클라이언트로의 전역 최적해 이주를 담당하고, 클라이언트의 유전자 알고리즘은 전역 최적해를 개체군에 포함시켜 지역 최적해를 개선한다. 서버는 염색체를 클라이언트에 할당하고 클라이언트에서 전송된 지역 최적해들을 개체군으로 하여 유전자 알고리즘을 수행한다. 클라이언트는 서버로부터 염색체를 할당받아 지역 최적해를 생성한 후 이를 서버에 전송한다. 서버의 유전자 알고리즘 수행결과 생성된 전역 최적해는 다시 클라이언트에 전달되어 지역 최적해를 개선하게 된다. 서버에서 유휴시간을 이용하여 클라이언트로부터 전송된 지역 최적해를 개선함으로써 최적해의 수렴속도가 향상된다.[14]

IV. 네트워크기반 병렬 유전자 알고리즘을 적용한 중앙집중형 동적부하균등기법

네트워크기반 병렬 유전자 알고리즘을 이용하여 중앙 스케줄러의 탐색능력을 향상시키면 부하균등기법의 태스크 완료 시간을 단축시킬 수 있고 평균 프로세서 이용률을 증가시킬 수 있다.

4.1 시스템의 구성

제안한 동적부하균등 시스템은 그림 2와 같이 중앙 스케줄러(CS), 각 프로세서 P_i, P_i에 할당된 태스크 큐(TQ_i)로 구성된다. 작업관리자 TM와 네트워크기반 병렬 유전자 알고리즘의 서버로 중앙 스케줄러를 구성하고 기존의 네트워크를 구성하는 프로세서들로 네트워크기반 병렬 유전자 알고리즘의 클라이언트들을 구성한다. 이렇게 함으로써 기존의 네트워크 망을 이용하여 구현할 수 있고 비용도 절감된다.

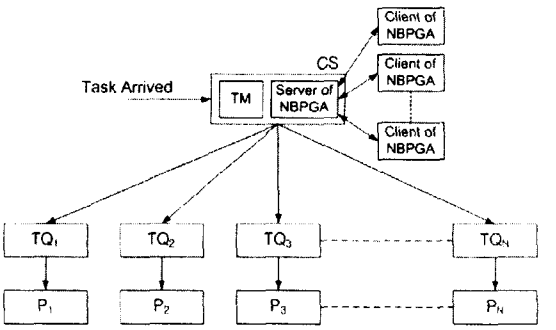


그림 2. 중앙집중형 동적부하균등시스템
Fig. 2. The centralized dynamic load-balancing system

4.2 중앙 스케줄러의 설계

각 프로세서의 태스크 처리상태와 큐의 상태를 관

리하고 슬라이딩 윈도우 기법을 이용하여 유전자 알고리즘의 염색체를 생성하는 중앙 스케줄러는 그림 3과 같이 작업관리자(Task Manager)와 네트워크기반 병렬 유전자 알고리즘으로 구성된다. 작업관리자는 작업할당관리자(Task Assignment Manager), 태스크 큐 및 염색체 인코더·디코더(Chromosome Encoder and Decoder)로 구성된다.

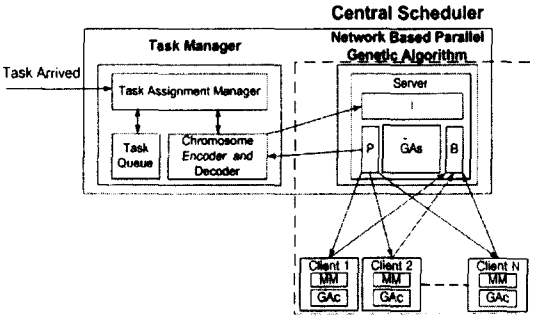


그림 3. 중앙 스케줄러의 구성
Fig 3. The composition of central scheduler

작업할당관리자는 시스템에 도착한 태스크들을 태스크 큐에 저장한 후 각 프로세서에 할당된 태스크 큐의 상태와 각 프로세서의 상태를 모니터링하여 작업할당 시점을 결정하고 작업할당이 필요할 때 염색체 인코더에 염색체의 인코딩을 명령한다. 염색체 인코더는 태스크 큐에서 대기중인 작업들을 슬라이딩 윈도우기법으로 인코딩하여 유전자 알고리즘에서 사용될 염색체를 생성하고 이를 네트워크기반 병렬 유전자 알고리즘의 서버로 전달한다. 유전자 알고리즘은 작업관리자의 명령에 따라 최적 작업할당을 탐색한다. 네트워크기반 병렬 유전자 알고리즘을 이용해 해공간을 분할하여 동시에 탐색이 가능하므로 고속으로 최적 작업할당을 얻을 수 있다.

4.3 네트워크기반 병렬 유전자 알고리즘의 적용방법

중앙 스케줄러의 최적작업할당 탐색에 네트워크기반 병렬 유전자 알고리즘을 적용하기 위해서 서버의 역할은 중앙 스케줄러가 담당하고, 클라이언트는 기존의 네트워크환경에 있는 프로세서들을 이용한다. 이렇게 함으로써 병렬컴퓨팅환경을 구축하지 않고 병렬 유전자 알고리즘을 구현할 수 있다.

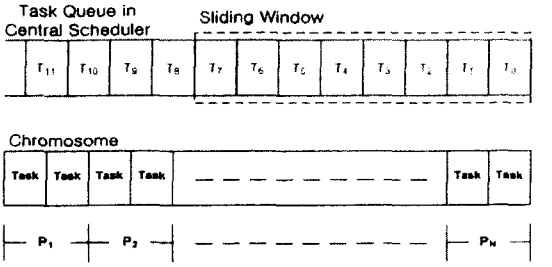


그림 4. 슬라이딩 윈도우 기법을 사용한 염색체의 인코딩
Fig 4. The chromosome encoding using sliding window method.

네트워크기반 병렬 유전자 알고리즘에서 사용되는 염색체는 그림 4와 같이 슬라이딩 윈도우기법을 사용하여 인코딩된다. 이 때 할당될 작업들은 각 프로세서에 동일한 수가 할당되도록 한다. 인코딩된 염색체의 길이 L_c는 식 (1)과 같이 표현될 수 있다.

$$L_c = N_{TP} \times N_P \tag{1}$$

여기서 N_{TP}는 한 프로세서에 할당될 태스크의 수, N_P는 프로세서의 수를 나타낸다.

유전자 알고리즘에서 선택연산은 적합도에 비례하여 염색체를 선택하는 룰렛 휠 선택과 이전 세대의 최적해를 유지하는 엘리트 보존전략을 함께 이용한다. 교차연산은 염색체 내에서 동일한 값이 중복되는 것을 방지하기 위하여 주기 교차기법을 이용하고 돌연변이연산은 임의의 두 점을 택해 값을 교환하는 방법을 이용한다. 그리고 각 처리 프로세서들의 상태와 각 처리 프로세서의 큐 상태를 고려한 평가함수 f는 식 (2)와 같다.[4]

$$f = \frac{1}{T_M} \times U_{avg} \times \frac{N_Q}{N_P} \tag{2}$$

여기서 T_M은 시스템 내에 있는 프로세서의 작업완료시간 중 가장 큰 값, U_{avg}는 평균 이용률, N_P는 프로세서의 수, N_Q는 허용 가능한 태스크 큐의 수이다.

4.4 중앙 스케줄러의 동작

시스템에 도착한 태스크는 중앙 스케줄러에 의해 그림 5와 같이 처리된다.

```
// Scheduling for task assignment //
// FTA : flag to show the need of task assignment //
// NP : number of processors //
// isIdle(i) returns the state of i-th processor //
// isEmptyTQ(i) returns state of i-th processor's queue //
task_assignment()
begin
  FTA = 0
  // Checking the need of task assignment //
  for (i = 1; i <= NP; i++)
    if (isIdle(i) == 1 && isEmptyTQ(i) == 1)
      FTA = 1
    end if
  end for
  if (FTA == 1)
    // Initialization //
    Encode Chromosome using Sliding Window Method;
    Send Chromosome to Server of NBPGA;
    // Searching the optimal task assignment //
    Find Elite Chromosome by the NBPGA;
    // Assign the tasks to the processors //
    Decode Elite Chromosome;
    Assign Tasks to Processors;
  else
    // Input task into the queue of central scheduler //
    Input Task into the Queue in Central Scheduler;
  end if
end
```

그림 5. 중앙 스케줄러의 동작
Fig 5. The operation of central scheduler

V. 실험 및 고찰

네트워크기반 병렬 유전자 알고리즘을 이용하여 최적작업할당을 결정하는 방법의 성능을 평가하기 위해 부하균등기법의 유연성에 따른 태스크 완료시간과 평균 프로세서 이용률을 모의실험하였다. 본 논문에서의 방법을 기존의 유전자 알고리즘을 적용한 Zomaya의 방법과 비교하여 성능향상을 고찰하였다.

5.1 실험조건

Zomaya[4] 등은 동적부하균등에 유전자 알고리즘을 적용한 실험을 통해 처리 프로세서의 수를 5, 작업의 수를 100, 작업의 최대크기를 20으로 설정한 경우에 처리 프로세서들의 평균 이용률이 가장 우수하게 나타남을 보였다. 본 논문에서도 이 조건을 적용하여 모의실험하였다. 이 때 교배확률은 0.7, 돌연변이확률은 0.1로 설정하였는데 이 값은 일반적으로 다양한 응용에 적용되는 것으로 알려져 있다.[15] 그리고 평가척도는 부하측정의 오버헤드를 최소화하기 위해 전체 태스크 완료시간과 평균 프로세서 이용률의 두 가지를 이용하였다. 실험은 부하균등기법의 유연성에 따른 평가척도의 변화를 모의실험하고 측정하였다. 통신 오버헤드를 고려하여 통신시 지연되는 시간만큼 기존의 방법에 대해 실행 세대수를 증가시켰다.

5.2 부하균등기법의 유연성에 따른 성능평가

임계값은 프로세서가 과부하상태인지 저부하상태인지를 나타낸다. 임계값을 조절함으로써 부하균등기법의 유연성을 결정한다. 과부하 임계값(heavy threshold)은 각 프로세서에 대해 과부하 허용도를 나타내고 저부하 임계값(light threshold)은 저부하 허용도를 나타낸다. 과부하 임계값 T_H와 저부하 임계값 T_L은 식 (3)으로 구할 수 있다.[4]

$$T_H = H \times L_{avg}$$
$$T_L = L \times L_{avg} \quad (3)$$

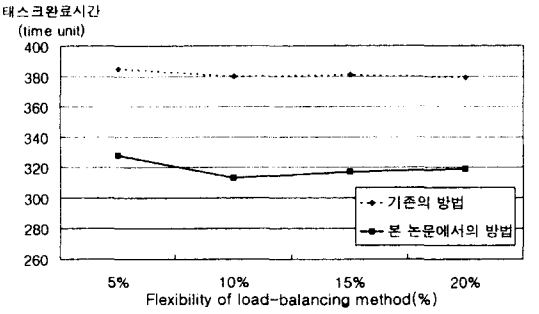
여기서 H는 과부하임계곱수, L은 저부하임계곱수, L_{avg}는 이전에 계산된 평균부하이다. H와 L값에 의해 부하균등기법의 유연성이 결정된다.

표 2. 부하균등기법의 유연성에 따른 H 값과 L 값
Table 2. H value and L value by flexibility of load-balancing method

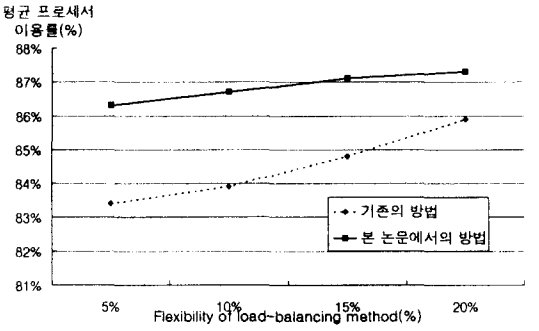
Flexibility of load-balancing method	H	L
5%	1.05	0.95
10%	1.1	0.9
15%	1.15	0.85
20%	1.2	0.8

예로 부하균등기법의 유연성이 5%인 경우 H는 1.05이고 L은 0.95이다. 이 경우 T_H와 T_L은 각각 이전에 계산된 평균부하의 1.05배와 0.95배이다. 실험에서 사용한 부하균등기법의 유연성, H 값 및 L 값은 표 2와 같다.

그림 6은 실험조건에 따른 모의실험 결과를 나타낸 것이다. 부하균등기법의 유연성이 5%와 20%인 경우에는 50 단위시간, 10%인 경우에는 70 단위시간, 15%인 경우에서는 60 단위시간 단축되었다. 평균 프로세서 이용률은 유연성이 5%인 조건에서 1% 향상되었으나 유연성이 20%까지 증가함에 따라 향상도가 감소하여 유연성이 20%인 조건에서 0.5% 향상되었다. 기존의 유전자 알고리즘을 적용한 방법과 비교하여 본 논문에서의 방법이 더 효과적임을 알 수 있다. 그러나 부하균등기법의 유연성이 증가함에 따라 평균 프로세서 이용률의 향상도는 감소한다. 그 이유는 부하균등기법의 유연성이 낮은 경우에는 작업할당계획의 성능이 평균 프로세서 이용률에 영향을 크게 미치기 때문이다. 이 경우 본 논문에서의 방법이 최적해의 수렴속도가 빠르기 때문에 더 효과적이다. 그러나 유연성이 증가하면 작업할당계획의 성능에 따른 영향이 감소하여 성능향상폭이 감소한다.



(a) 전체 태스크 완료시간
(a) total task completion time



(b) 평균 프로세서 이용률
(b) average processor utilization

그림 6. 부하균등기법의 유연성에 따른 성능 비교
Fig 6. The comparison of performance by flexibility of load-balancing method

VI. 결 론

중앙집중형 동적부하균등기법의 성능은 중앙 스케줄러의 최적작업할당 탐색능력에 따라 좌우된다. 본 논문에서는 중앙 스케줄러의 탐색능력을 향상시키기 위해서 네트워크기반 병렬 유전자 알고리즘을 이용하였다. 유전자 알고리즘의 서버는 중앙 스케줄러에 통합되고 클라이언트들은 기존의 네트워크환경에 연결된 프로세서들을 이용한다. 클라이언트들에서 최적작업할당을 분할하여 탐색하므로 처리속도가 향상된 작업할당계획을 수립할 수 있었다. 선택연산에 룰렛휠 방법과 엘리트 보존 방법을 함께 사용하여 최적해 수렴속도를 향상시켰다. 염색체는 슬라이딩윈도우기법을 이용하여 인코딩하였고, 교차연산과 돌연변이연산은 각각 주기교차방법과 전형적인 돌연변이방법을 이용하였다. 기존의 유전자 알고리즘을 이용한 방법과 비교 분석한 결과, 본 논문에서의 방법이 기존의 방법보다 전체 태스크 완료시간과 평균 프로세서 이용률이 향상됨을 확인할 수 있었고 최적해의 수렴속도가 빠르기 때문에 부하균등기법의 유연성이 낮은 경우에 보다 효과적인 방법임을 확인할 수 있었다.

참고문헌

[1] S.H. Bokhari, "On the Mapping Problem," *IEEE Trans. Computers*, vol.30, no.3, pp.550-557, Mar. 1981.

[2] S. Salleh and A.Y. Zomaya, *Scheduling in Parallel Computing Systems: Fuzzy and Annealing Techniques*. Kluwer Academic, 1999.

[3] A.Y. Zomaya, "*Parallel and Distributed Computing: The Scene, the Props, the Players*," Parallel and Distributed Computing Handbook, A.Y. Zomaya, ed., pp5-23. New York: McGraw-Hill, 1996

[4] A.Y. Zomaya and Y.H. Teh, "Observations on Using Genetic Algorithms for Dynamic Load-Balancing," *IEEE Trans. on Parallel and Distributed System*, vol.12, no.9, pp.899-911, 2001. 9.

[5] F. Seredynski, "Dynamic Mapping and

Load Balancing with Parallel Genetic Algorithms", *IEEE World Congress on Computational Intelligence., Proc. of the First IEEE Conference on*, vol. 2, pp. 834-839, 1994, 7.

[6] M. Munetomo, Y. Takai and Y. Sato, "A Genetic Approach to Dynamic Load Balancing in a Distributed Computing System", *IEEE world congress on computational Intelligence proc. of the first IEEE Conference on* Vol. 1, p.418-421, 1994, 7.

[7] W. A. Green, "Dynamic Load-Balancing via a Genetic Algorithm," *Tools with Artificial Intelligence, Proc. of the 13th International Conference on*, pp. 121-128, 2001. 11.

[8] F. Bonomi and A. Kumar, "Adaptive Optimal Load-Balancing in a Heterogeneous Multiserver System with a Central Job Scheduler," *IEEE Trans. Computers*, vol.39, no. 10, pp. 1232-1250, 1990. 10.

[9] Y. Lan and T. Yu, "A Dynamic Central Scheduler Load-Balancing Mechanism," *Proc. IEEE 14th Ann. Int'l Phoenix Conf. Computers and Comm.*, pp. 734-740, 1995

[10] H. C. Lin and C. S. Raghavendra, "A Dynamic Load-Balancing Policy with a Central Job Dispatcher(LBC)," *IEEE Trans. Software Eng.*, vol. 18, no. 2, pp. 148-158, 1992. 2.

[11] J.H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975

[12] K. Kojima, W. Kawamata, H. Matsuo and M. Ishigame, "Network based Parallel Genetic Algorithm using Client-Server Model", *Proc. of CEC2000*, p.244-249, 2000.

[13] K. Kojima, H. Matsuo and M. Ishigame, "Reduction of Communication Quantity for Network Based Parallel GA", *Proc. of the CEC2002 Congress on* , Volume: 2, p.1715-1720, 2002. 5.

[14] 송봉기, 김용성, 성길영, 우종호 "서버의 계산 능력을 활용한 네트워크기반 병렬 유전자 알고리즘의 성능향상", 대한전자공학회 논문지 제 41권 CI편 제4호, pp385-390, 2004. 7.

[15] J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transaction on System, Man and Cybernetics*, Vol.SMC-16. No. 1, 1986. 1.

저자 소개

송봉기 (Bong-Gi Song)

1997년 부경대학교 전자공학과 학사 졸업.
1999년 부경대학교 정보시스템 석사 졸업
2005년 부경대학교 정보시스템 박사 졸업.
※관심분야 : 유전자알고리즘, 마이크로프로세서 등

성길영(Kil-Young Sung)

1996년-현재 : 경상대학교 정보통신공학과 교수, 해양산업연구소 연구원
제 8 권 제 5 호 참고.

우종호(Chong-Ho Woo)

1996-현재 : 부경대학교 전자컴퓨터정보통신 공학부 교수
제 8 권 제 5 호 참고.